Using Java analysis studio as an inferface to the ATLAS offline framework

J. Hrivnac

Using Java Analysis Studio as an interface to the Atlas Offline Framework

J.Hrivnac, LAL, Orsay, France

Abstract

Huge requirements on computing resources have made it difficult to run Frameworks of some new HEP experiments on the users' personal workstations. Fortunately, new software technology allows us to give users back at least a bit of the user-friendliness they were used to in the past. A Java Analysis Studio (JAS) plugin has been developed, which accesses the Python API of the Atlas Offline Framework (Athena) over the XML-RPC layer. This plugin gives a user the full power of JAS over the resources otherwise only available within Athena. A user can access any Athena functionality and handle all results directly in JAS. Graphical adapters to some Athena services have been delivered to ease the access even further.

ATHENAEUM

Athenaeum[1] is a Java Analysis Studio (JAS)[2] plugin which allows to access (remote) XML-RPC[3] Server. It is currently used to access Python Server running as a part of the Atlas offline Framework - Athena[4]. Any Python script can be send directly from JAS to Athena. Results (usually in XML) are send back and can be processed within JAS. Special Python scripts are provided to automatically present some standard Athena data within JAS. See Fig. 1 for the overview of the Athenaeum Architecture.

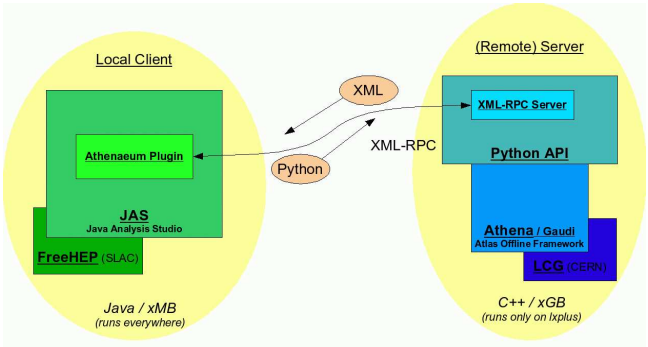

Figure 1: Athenaeum Architecture.

Connection between Athenaeum plugin and Athena server is made over encrypted XML-RPC channel. Python Server running inside Athena has been implemented by the Atlantis[5] team, it can be easily started at any moment of the Athena Python process.

Remote Script

Any Python script can be send from Athenaeum to the running Athena. Its results (either a plane text or an XML fragment) are automatically send back and can be processed using the standard JAS mechanisms.

User can mix Python running within JAS and Python running in a (remote) Athena. Athena Python scripts could be moved to JAS.

Record Source

JAS with Athenaeum plugin can steer Athena Event Loop, interpreting Athena as a *Record Source*. Processing functions or classes can be then written which are called for each event delivered by Athena.

REMOTE PROXY

For some specific Athena data structures, Proxies have been created to provide their customized automatic presentation. Those Proxies are in general implemented by:

- Athena Python script to extract data from Athena.
- JAS wrapper to present/handle data inside JAS.
- XML schema to describe data.

See Fig. 2 for the overview of the Proxies Design.

Because implementing pre-defined interfaces from Athenaeum, those Proxies will make themselves automatically available inside JAS system in an organic way. They will, for example, appear on the appropriate menus.

XML SCHEMA REPRESENTATION

Because the same data are shared between different Frameworks/Applications implemented in different languages (Java, Python, C/C++), its common definition is needed. An XML Schema has been written describing some common Athena data structures. All data representations are then automatically derived (generated) from that XML Schema. The whole mechanism is described on the Fig. 3.

INTERACTION WITH COOL

Cool[6] is a Conditions Database developed by CERN LCG[7] project. It is available only via its proprietary API which makes access to it from independent applications (like JAS) difficult. Athenaeum offers a way to access

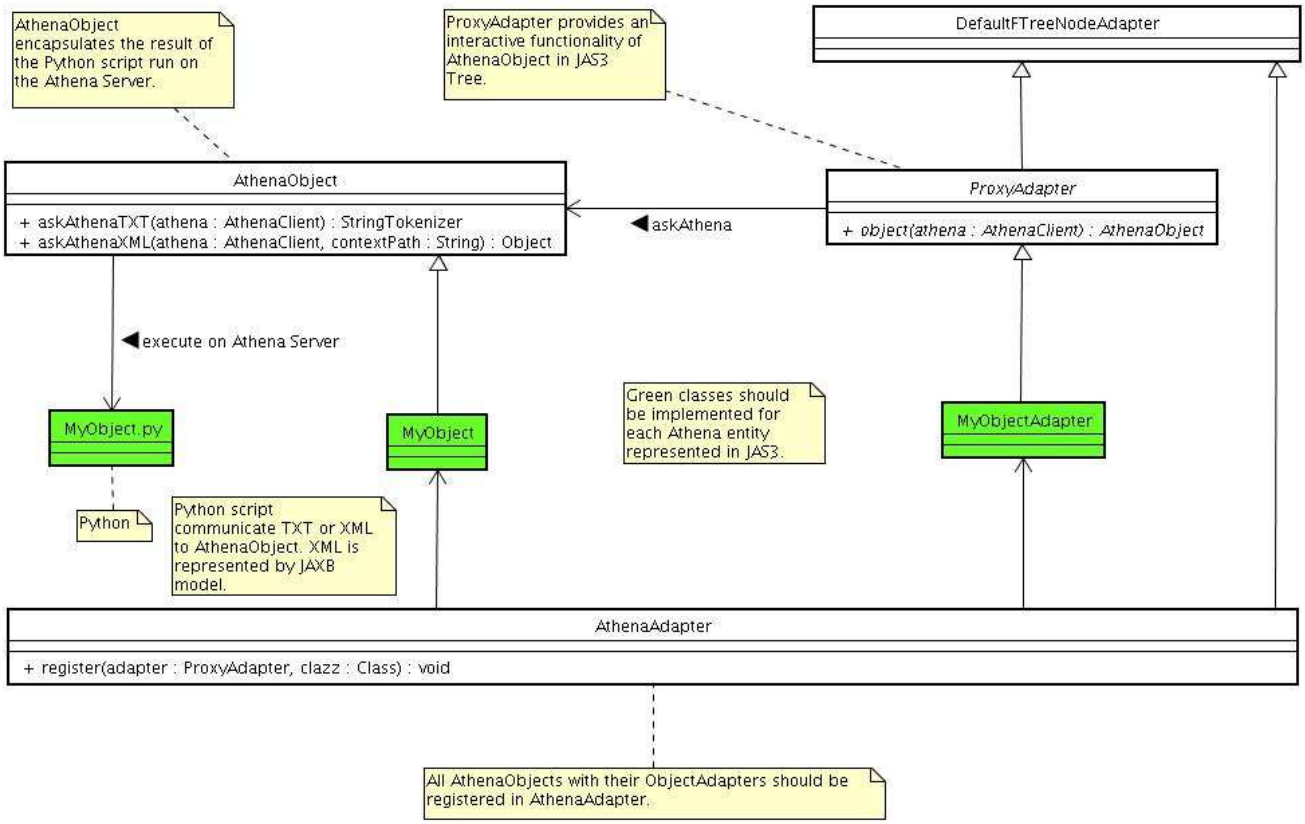

Figure 2: Construction of Proxy.

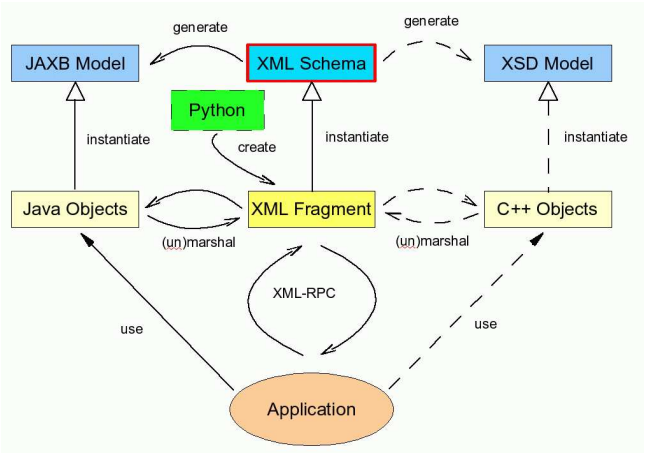

Figure 3: XML Schema Representations.

Cool databases using Athena Python component. Specific Athenaeum proxies have been written to make that connection automatic.

See Fig. 4 for an example of the running Cool Browser.

Cool data are received in an XML form. They can be then represented in several ways:

- Directly as XML fragments.
- As derived Objects of several kinds.
- As a graphical Tree of elements.
- As a set of (AIDA)[8] NTuples.
- As an HTML page created using XSLT stylesheet.

All those data can be accessed via JAS Graphical Interface, using scripting interface in several languages (Java, Python, Pnuts) or directly from Java or Python.

DISTRIBUTED INTERACTIVE ENVIRONMENT ARCHITECTURE PROJECT

Athenaeum is a component in a planned *Distributed Interactive Environment* (see Fig. 5). In this environment, user will interact with the Framework using lightweight client (JAS) and all data will accessed using a network of Virtual Servers connecting to actual databases. All data will be described by the application and language neutral XML Schema.

Other components of the Architecture are SQLTuple[9]/ColMan[10] and Sequoia[11].

ARCHITECTURE ADVANTAGES

Presented Architecture has many advantages:

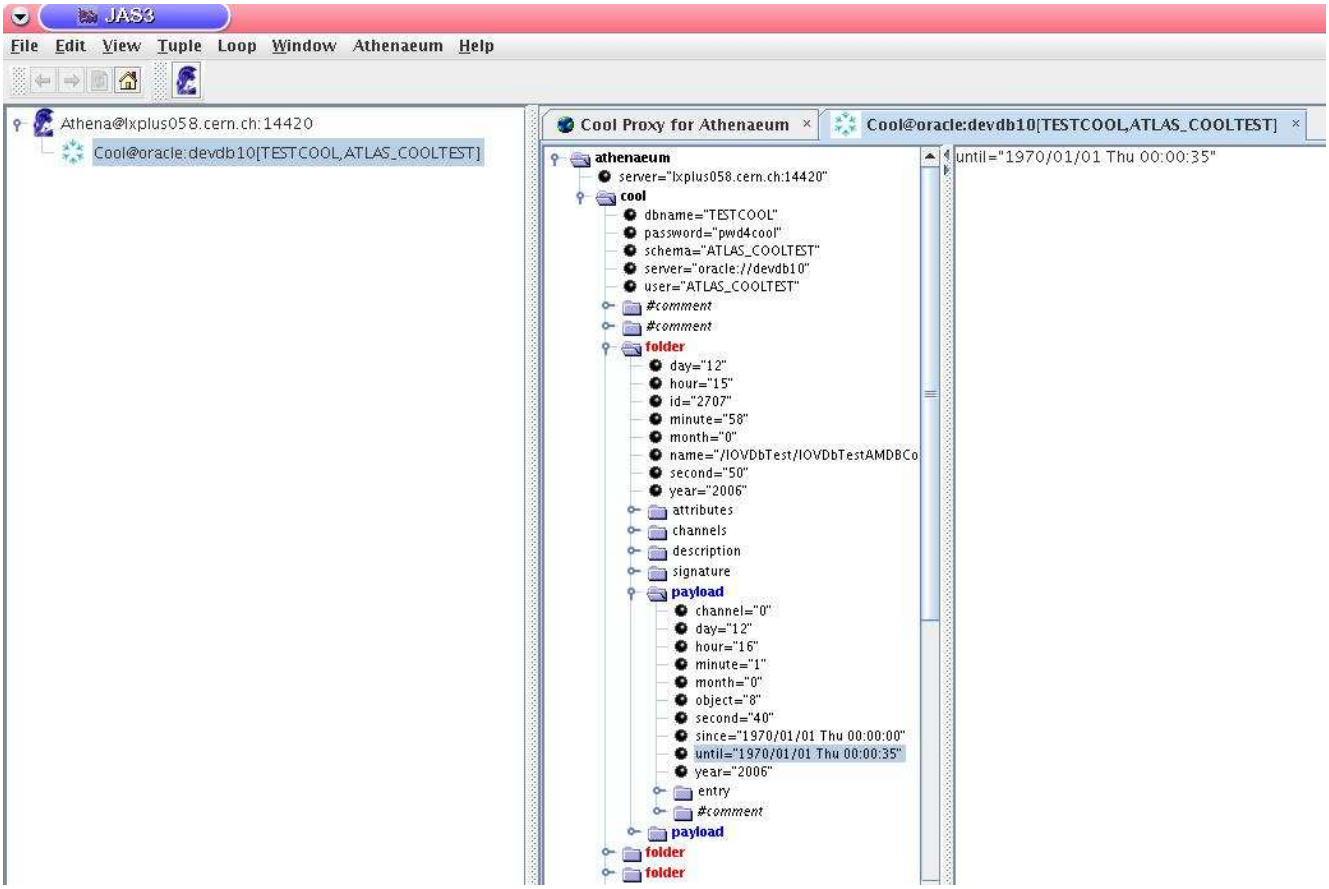

Figure 4: CoolBrowser.

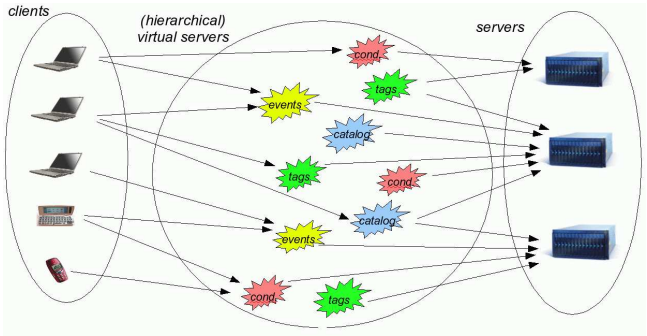

Figure 5: Distributed Interactive Environment Architecture Project.

- Light local client (JAS): It runs on any release of any platform. It offers fully interactive GUI, scripting and API in several languages. It is easily extensible by modular plugins.

- Server on a powerful machine: It can be close to data, replicated and hierarchised when useful.

- Standard communication protocols: Efficient XML-RPC connection is used for the Control Flow and small data. More performant protocols (like JDBC[12], xrootd[13],...) can be used if bigger data flow is needed.

PROBLEMS

Interacting with framework based on proprietary interfaces poses several problems. In the case of connection on LCG-based Athena, following limitations are most serious:

- Python API to Athena is incomplete. Only a subset of C++ Athena API is available via Python. The API is undocumented. The C++ documentation created by Doxygen[14] is not enough for documentation of its Python API. It is not easy to guess the meaning of weakly-typed methods. Available code fragments on Web/Wiky are often out-of-date. The API is also very unstable, there are too many change too often.

- No abstract data definition is available. The actual data model is hidden very deep in the C++ header files forest. Athenaeum XML Schema has been written for data passed around. XML, Java, Python and C++ incarnations can be created from them.

PLANS

The work on the Athenaeum will continue. Several subjects will be attacked in the near future:

- Athenaeum will be generalized to be usable with other Monolithic Frameworks within HEP.
- Lazy and Compressed data transport will be supported to speed operations up.
- User will be able to supply customized XSLT stylesheet.
- More standard Proxies will be delivered.
- A possibility of the Athena startable remotely from Athenaeum will be investigated.
- A deployment of Athena Servers network will be studied.

REFERENCES

[1] http://home.cern.ch/hrivnac/Activities/Packages/Athenaeum, https://uimon.cern.ch/twiki/bin/view/Atlas/HowToUseJAS

[2] http://jas.freehep.org/jas3

[3] http://www.xmlrpc.com

[4] http://atlas.web.cern.ch/Atlas/GROUPS/SOFTWARE/OO/architecture/General

[5] http://atlantis.web.cern.ch/atlantis

[6] http://lcgapp.cern.ch/project/CondDB

[7] http://lcgapp.cern.ch

[8] http://aida.freehep.org

[9] http://home.cern.ch/hrivnac/Activities/Packages/SQLTuple

[10] http://home.cern.ch/hrivnac/Activities/Packages/ColMan

[11] http://sequoia.continuent.org/HomePage

[12] http://java.sun.com/products/jdbc

[13] http://xrootd.slac.stanford.edu

[14] http://www.stack.nl/ dimitri/doxygen